

# 纯Web离线化以及更新方案

于秋



于秋  
智能支付—前端技术专家

8月份加入美团

11月开始离线化方案设计

目前负责组内的前端基础技术服务

# 从哪里来？

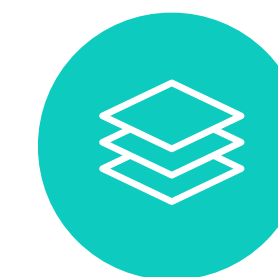


为满足现今用户的消费习惯，同时作为线下POS收单渠道的补充，我们添加了一个二维码扫码付的产品。具备微信支付、支付宝支付、支付后评价和会员营销等功能，目标定位于多码合一，具有快速收银、低成本、统一管理的特点和优势。



快速收银

吧台扫码即可付款



支付通道多

微信，支付宝，美团App

## 先天不足

支付容器

网络场景

业务迭代

信任感

# 是谁？

如果是你，你会怎么做，来解决这些问题



## 2B 青年

不要跟我说问题，都是你的问题



## 普通青年

各种缓存上呀！！



## 文艺青年

不要在意这些细节



# 高手高手 高高手



当我们打开一个网站的时候，到底发生了些什么？  
在这个漫长的链路上面，我们能做什么？

## Http请求

### 少并且小

减少Http请求的数量，请求的体积大小，比如Cookie的大小。

## Http链接

### 越快越好

越快建立起Http链接越好，中间DNS解析，tcp握手都是我们可以优化的点。

## 加载页面

### 下载资源

利用2的原则，我们应该尽量少的发Http请求，比如图片，字体以及Js文件等等。

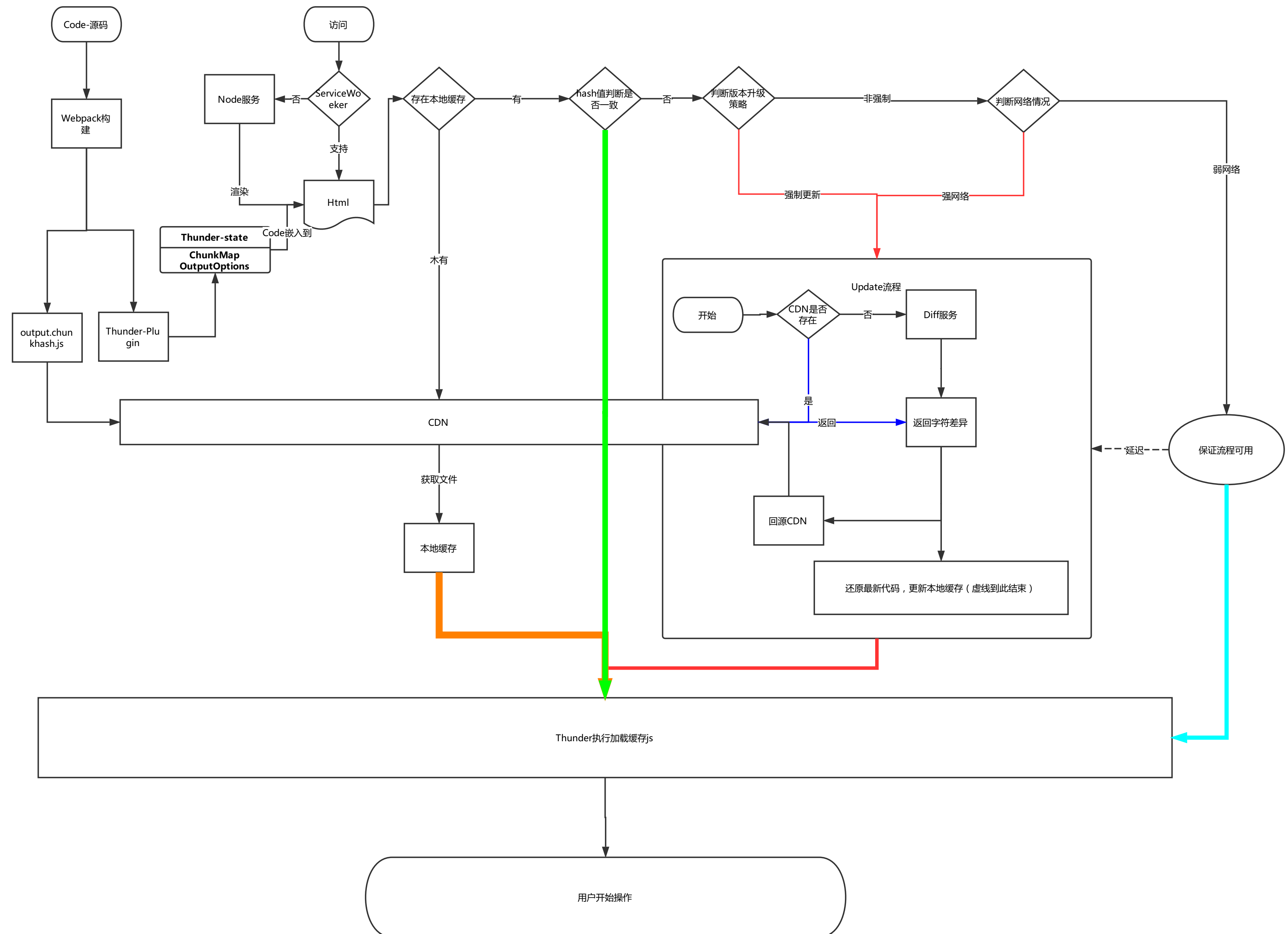
## 解析Html

### 页面渲染

dom 树解析、js执行和首屏渲染是串行的。所以我们会把Js放到Body后面。

很长，怕不怕？

# 到哪里去？



撸猫要从头开始

## 重要阶段

A

### 构建阶段

构建时我们需要拿到什么？

哪些可以省略？

哪些可以扩展？

B

### 访问阶段

什么条件会影响方案执行？

条件是否唯一？

条件是否可靠？

C

### 更新策略

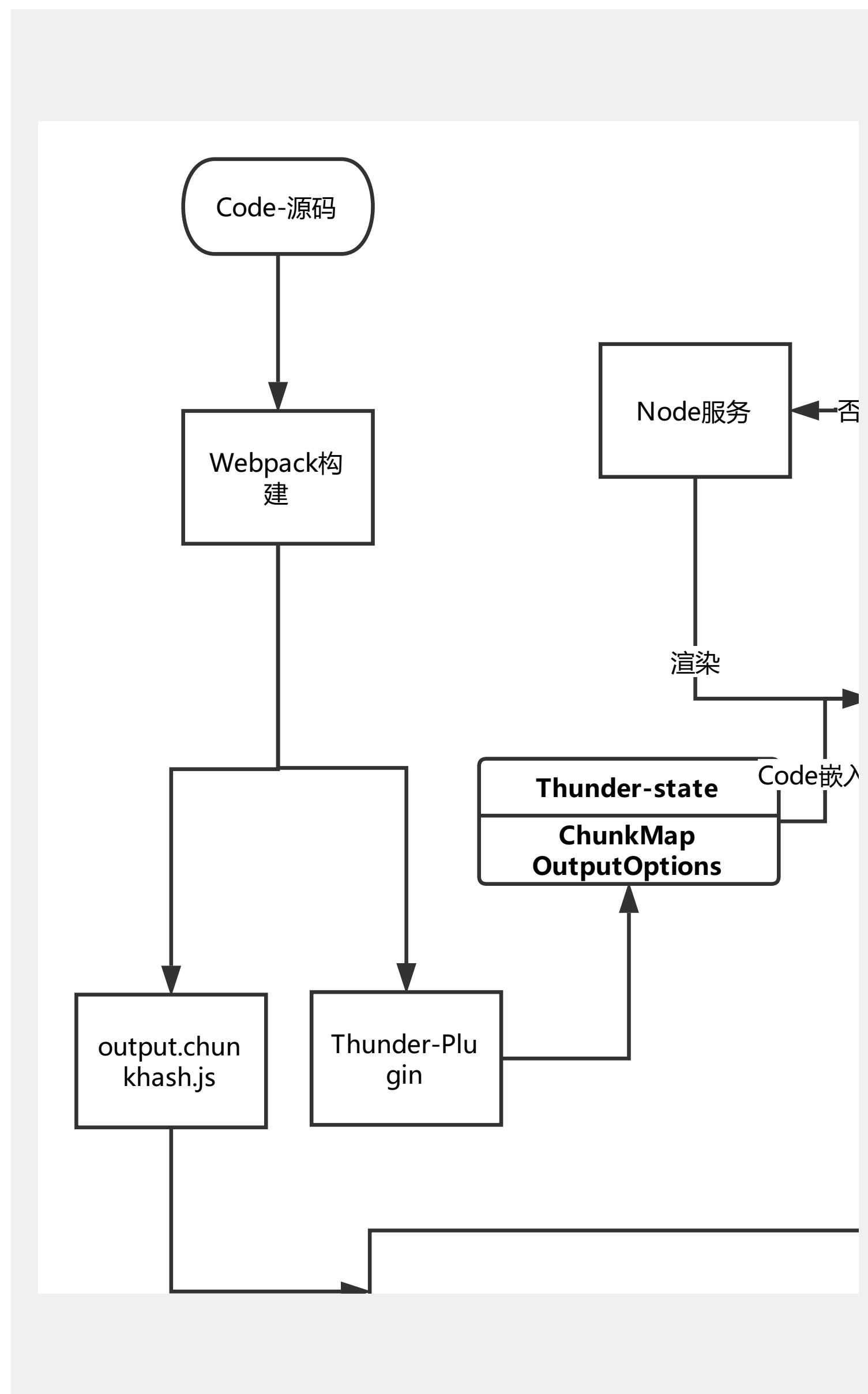
字符比较还是全量返回？

从哪里去结果？

能否减轻压力？



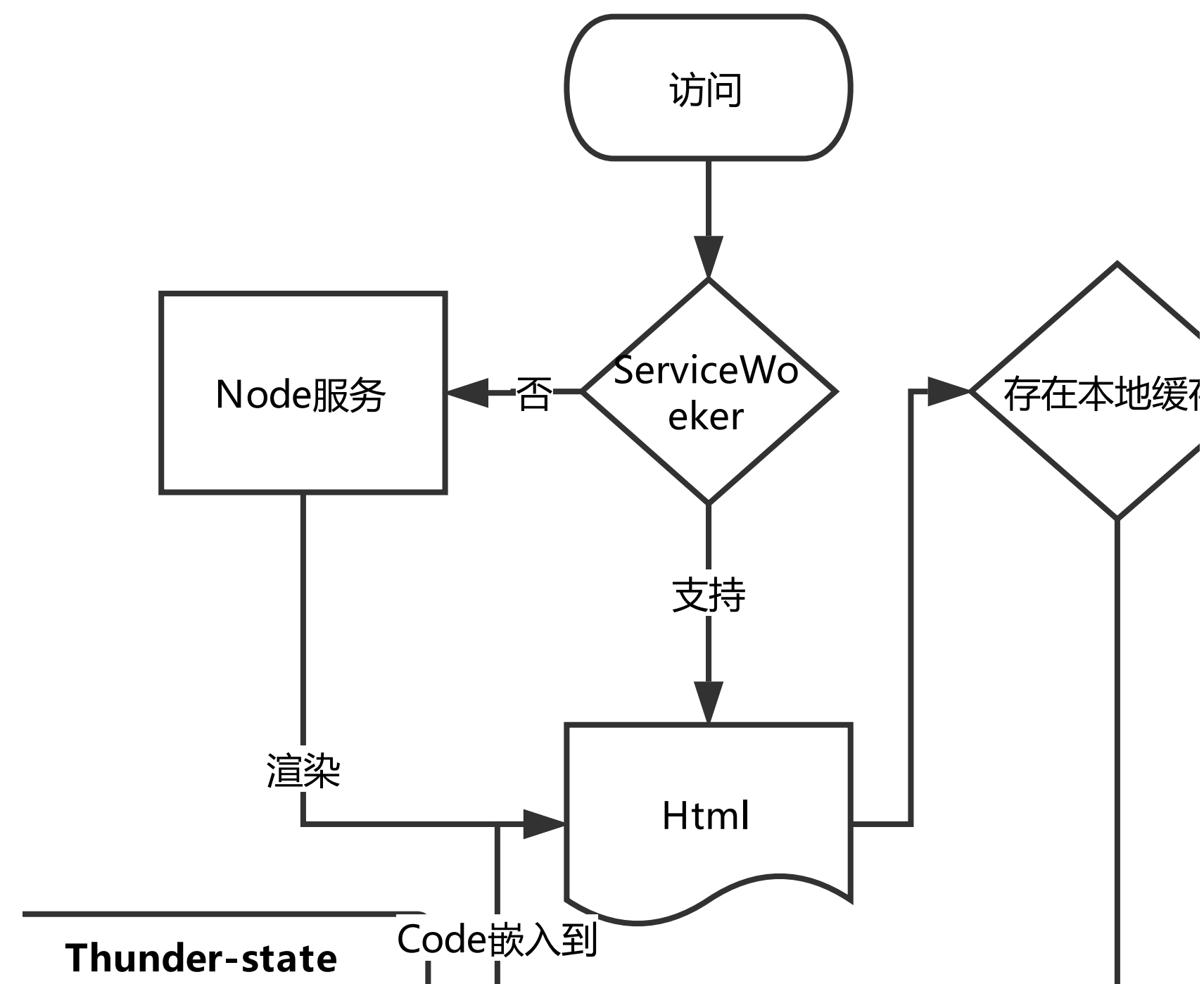
## Webpack插件 信息源



- 01 ChunksMap (hash值)
- 02 BSOpts (Diff服务的配置信息)
- 03 ServiceWorker (PWA)
- 04 History (版本信息)
- 05 NetworkTactic (网络信息)

PWA—渐进式网页

# Service Worker



注册

通信

控制

跳坑和爬出来

## 实践经验

SW中GlobalScope 的 Navigator 的 UserAgent 和 封装过的Browser 的UA是不一致的

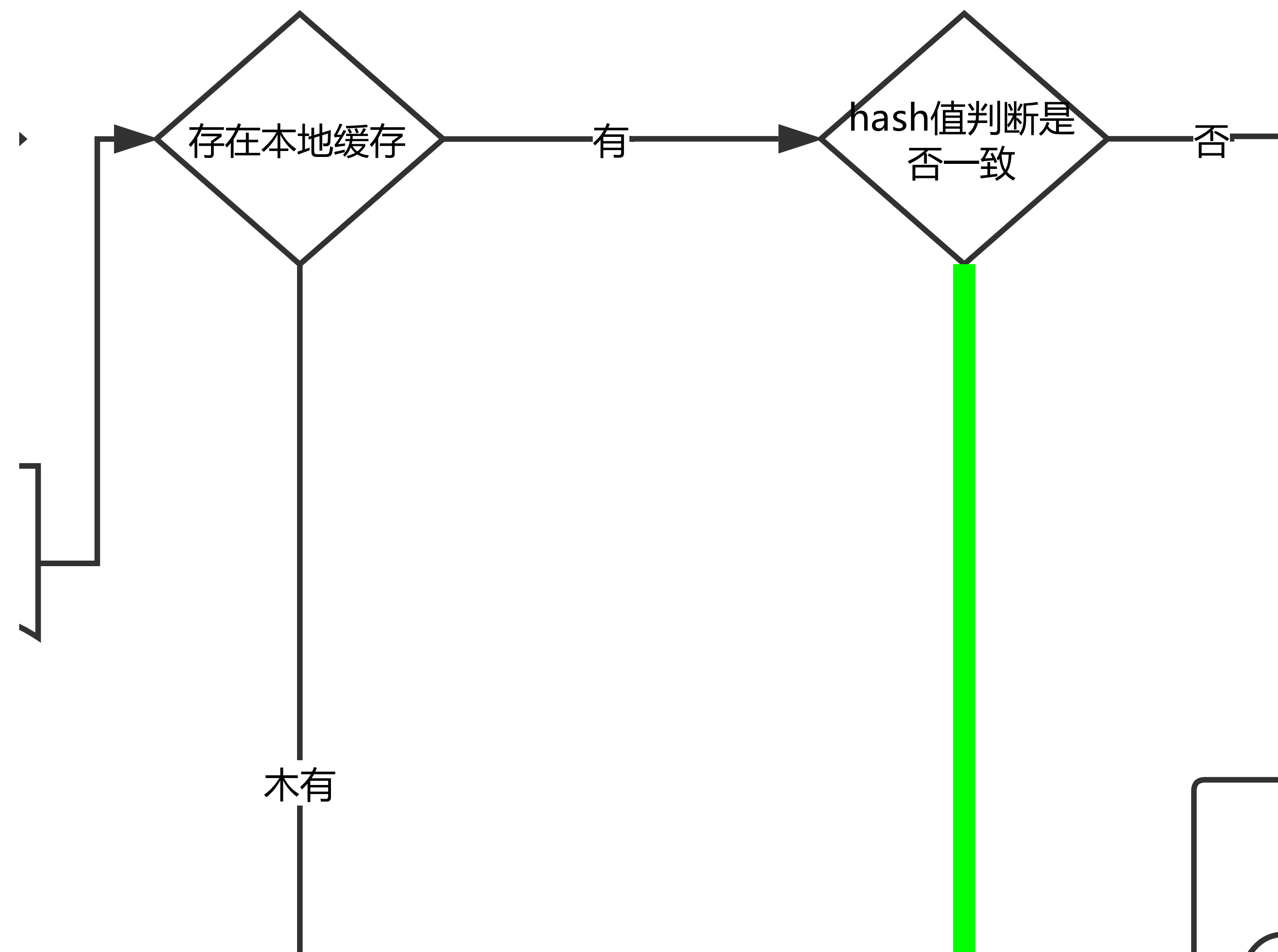
通过 SW缓存的资源不能是 http 下的资源，在 https 下加载 http 资源会引发 Mixed Content，导致资源不能被正常加载。如果有不能避免的 http 内容引入，尝试在 excludes 中加入 '^http:\\\\V'。

```
app.get('route-path/service-worker.js', (req, res) => {  
  res.set({  
    'Content-Type': 'application/javascript; charset=UTF-8',  
    'Service-Worker-Allowed': '/your-scope',  
  })  
  res.send(serviceWorker)  
})
```



本地缓存一致性

# Local Storage



# 本地缓存情况

三种情况

hash一致

休息休息

业务没有迭代的情况下，用户再次访问我们产品时，hash值是一致的，所以不用更新，直接本地读取执行即可。

无缓存

首次访问

首次访问我们业务时，本地是没有缓存，所以我们走平常的加载情况，全量拉取。

hash不同

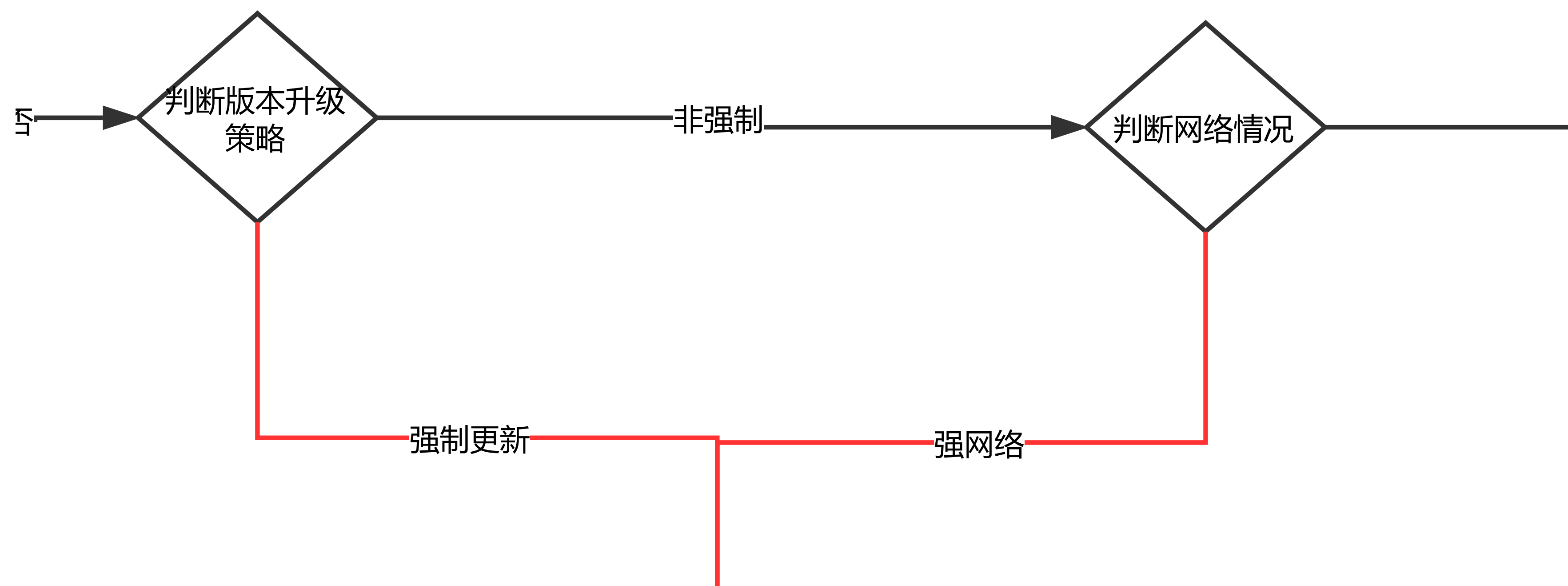
业务迭代

如果发布了新的版本，一些文件的hash值必要不一致，我们就要进行下一步，准备获取这些不一致文件的Diff结果了。



用户体验为中心

## 版本和网络策略



# 版本和网络情况

前端有些特点，以用户体验为中心

强制

有必修的BUG



BUG

非强制

体验优化



UI

弱网

网络差的很



2G

强网

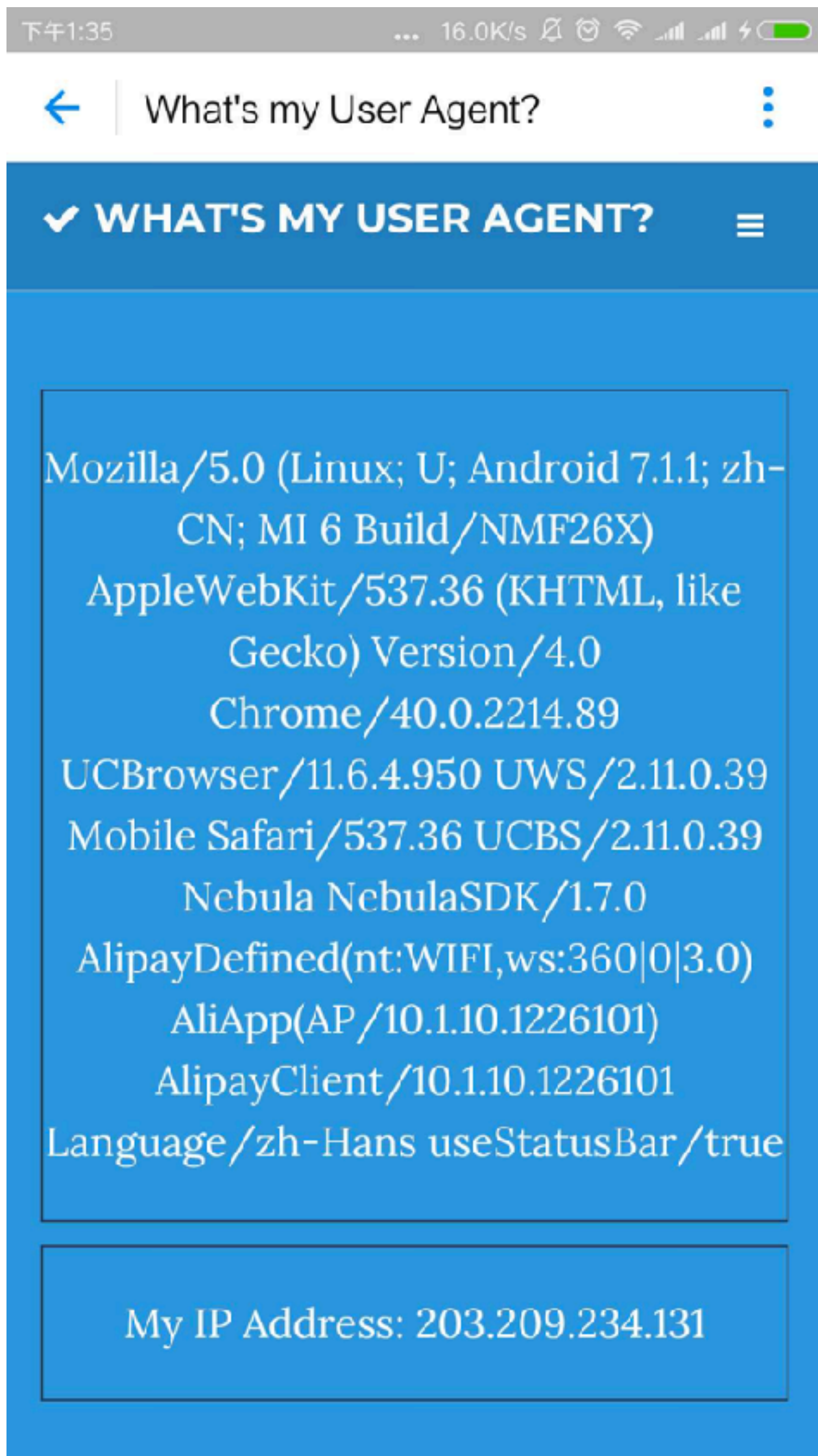
专属光缆



wifi

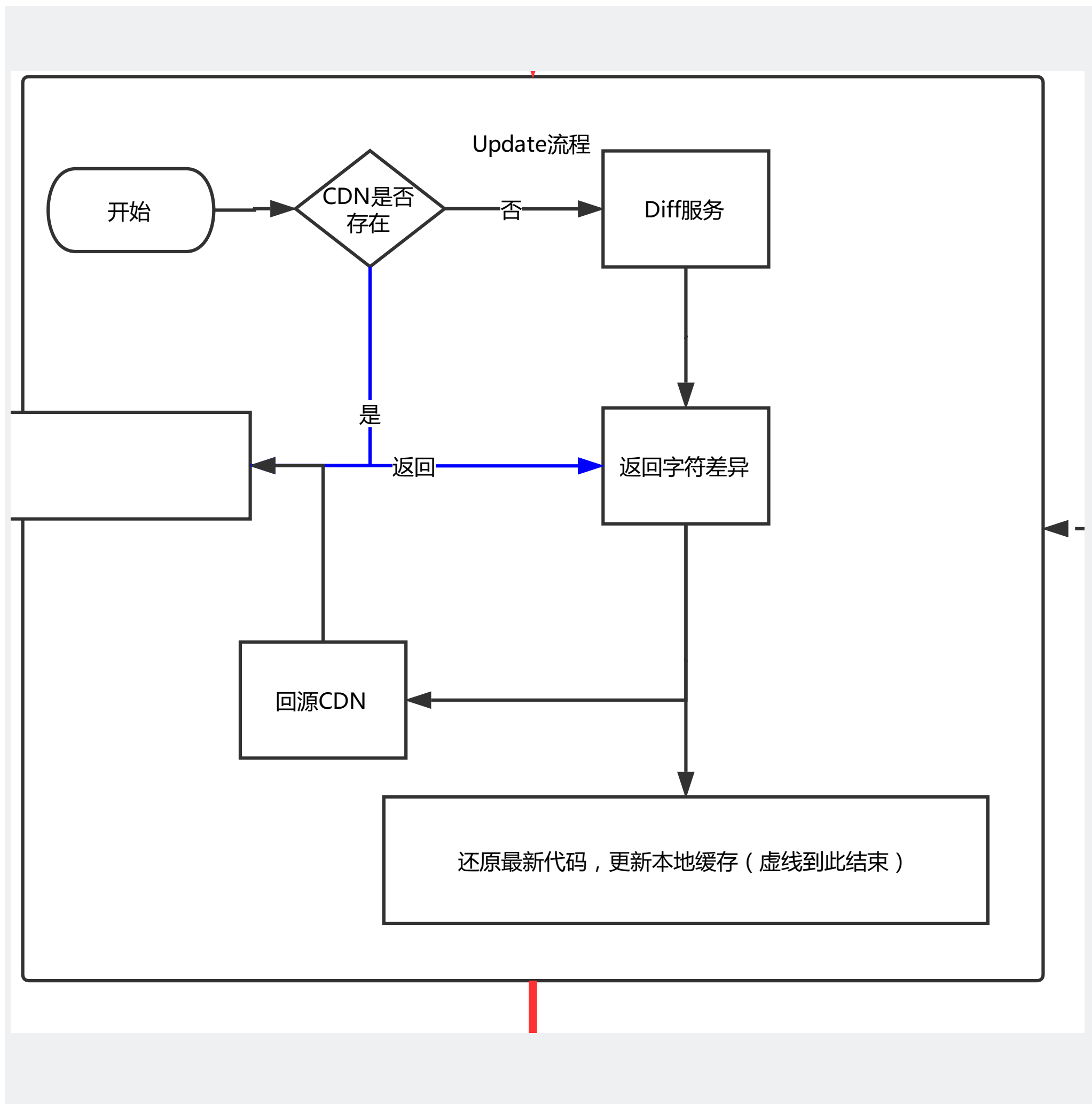
如何获取网络情况

## 网络情况



字符级更新

## 更新策略



CDN

计算Patch

还原文件

古老而优美

## Diff算法

### Myer's diff algorithm

用最小的“编辑步数”将字符串转换

平均算法复杂度 $O(|u|+|v|+d^2)$

将数百KB的diff计算，从分钟级减少到亚秒级



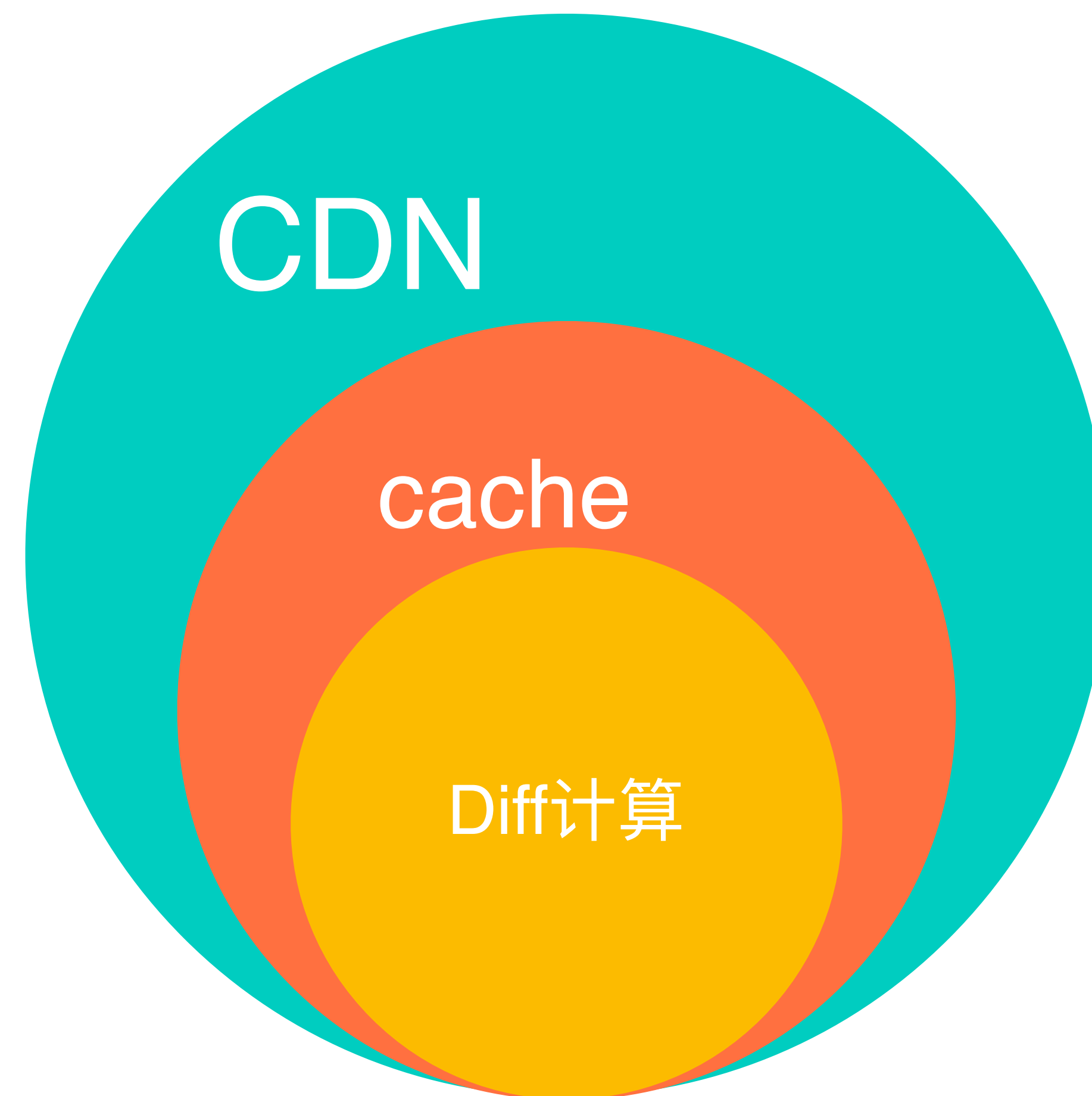
高流量高并发

稳定

善用基础设施

减少计算频率

避免习惯误区



回忆杀

## 回顾总结

### 01 构建阶段

我们在构建阶段会拿到各种配置信息，以供后续使用：hash值、网络策略开发、Diff服务配置信息、版本信息等

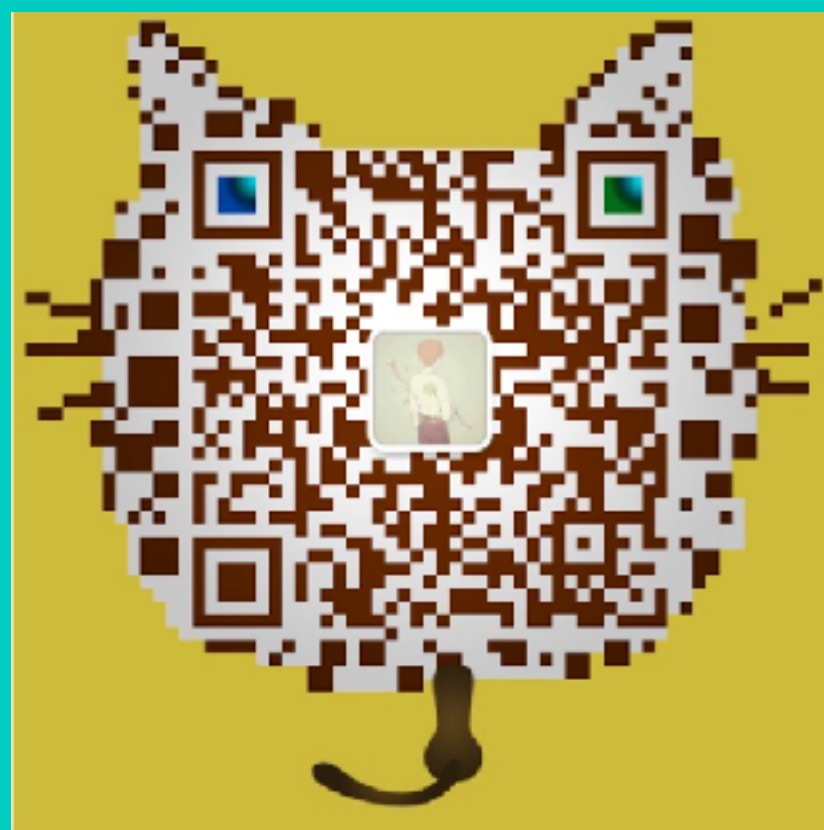
### 02 访问阶段

我们在这个阶段经过一系列的条件分支判断，根据判断的结果走不同的分支执行不同策略。总体上归结于：1、是否需要更新？2、先更新还是先执行？

### 03 更新策略

获取同一个业务文件的不同版本的patch，然后再还原到最新的版本，需要考虑大业务量的时候服务的压力。这个适合快速迭代业务的时候，减少带宽流量，加快文件拉取速度，提升弱网体验。

数据效果： 30%+ \*\*机密



扫码关注我，加我微信。

招聘：前端开发岗位

邮箱：[yuqiu@meituan.com](mailto:yuqiu@meituan.com)

# Q&A